LU: 349/16

Nro de Orden: 274	Ejercicio 1	Hoja 1 Eric Brandwein

Si $m < \frac{d \cdot (d+1)}{2} \overset{?}{\Rightarrow} \exists v \,/\, d(v) < d$.

1	2	3	4	5
2	2	2	1,4	1,6

Por lo visto en clase, sabemos que

$$m = \left[\sum_{v \in V} d(v)\right] / 2$$

siendo V el conjunto de nodos de G. Si $m < \frac{d \cdot (d+1)}{2}$, nos queda que

$$\left[\sum_{v \in V} d(v)\right] / 2 < \frac{d \cdot (d+1)}{2}$$

$$\Updownarrow$$

$$\sum_{v \in V} d(v) < d \cdot (d+1)$$

~~También sabemos que si un grafo tiene n nodos entonces tiene como máximo $\frac{n(n-1)}{2}$ aristas, lo que nos dice que si un grafo tiene m aristas entonces tiene por lo menos~~

~~$2m \leq n(n-1)$~~

De acá podemos dividir en tres casos:

1. $|V| > (d+1)$: Si sucede esto, el promedio de los grados de V será menor a $\frac{d \cdot (d+1)}{|V|} < d$, y como el promedio de los grados es menor a d, no pueden ser todos los grados mayores a d, y por lo tanto hay por lo menos un vértice con $d(v) < d$.

2. $|V| = (d+1)$: De nuevo, el promedio de los grados será menor a $\frac{d \cdot (d+1)}{|V|} = \frac{d \cdot (d+1)}{(d+1)} = d$, y por ende el promedio será menor a d, lo que hará que al menos un nodo tenga grado menor a d.

3. $|V| < (d+1)$: Como la cantidad de nodos es menor a $d+1$,

sigue

o sea, ~~como máximo~~ menor o igual a δ (por $\delta \in \mathbb{N}$), cada nodo podrá "conectarse" como máximo a $\delta - 1$ otros nodos, o, lo que es lo mismo, tendrá como máximo grado $\delta - 1$. Como $\delta - 1 < \delta$, todos los nodos tendrán grado $< \delta$, y entonces habrá por lo menos uno que tenga grado $< \delta$.

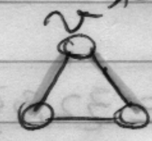
a) Siendo K_n el grafo de n nodos que posee todos las aristas posibles, para $n \in \mathbb{N} - \{0, 1\}$, K_n es un grafo junta.

~~Para K_1 K_1 no puede separarse en~~ Los nodos de K_1 no pueden separarse en dos subconjuntos no vacíos, porque ~~tiene~~ hay uno solo. Por lo tanto, el conjunto de nodos de K_1 no puede ser una unión de dos subconjuntos no vacíos, y entonces K_1 no es un grafo junta.

Para ver que K_n con $n \in \mathbb{N}$, $n > 1$ es un grafo junta, tomaremos uno de sus nodos y lo llamaremos v. Sabemos que v ~~esta~~ tiene una arista por cada otro nodo en K_n que hace que sean adyacentes. Por lo tanto, si tomamos $G_1 = K_n - v$ y $G_2 = \{v\}$ su grafo junta será K_n, porque ~~$V_{K_n} = V_1 \cup V_2$~~ $V_{K_n} = V_1 \cup V_2$, K_n tiene todos los ejes de G_1 y G_2, y todos los ejes que conectan al único nodo de G_2 con todos los nodos de G_1 están en K_n.

b) C_n es un grafo junta si y sólo si $n=3$ ó $n=4$

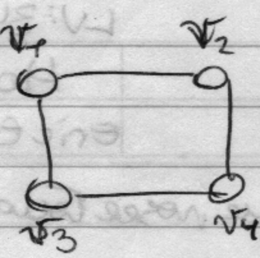
$\boxed{n=3 \Rightarrow C_n \text{: grafo junta}}$ Si agarramos un nodo de C_3 y lo llamamos v, veremos que v posee una arista por cada uno de los nodos restantes que la conecta a ellos.



Por lo tanto, si tomamos $G_1 = \{v\}$ y $G_2 = C_3 - v$, el grafo junta de G_1 y G_2 será C_3.

~~C_n: grafo junta $\Rightarrow n=3$) n no puede ser menor a 3, por definición de C_n.~~

$\boxed{n=4 \Rightarrow C_n \text{: grafo junta}}$ Al dorso de la hoja está dibujado el grafo C_4.



Si separamos a C_4 en los grafos de aristas $\underbrace{\{v_2, v_3\}}_{G_1}$ y $\underbrace{\{v_1, v_4\}}_{G_2}$,

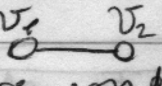
veremos que cada nodo de cada subgrafo se conecta a todos los otros nodos del otro subgrafo por medio de una sola arista de C_4. Si tomamos estos subgrafos como G_1 y G_2, llegaremos a que su grafo junta es C_4.

$\boxed{n \notin \{3,4\} \Rightarrow C_n \text{ no es grafo junta}}$ Si $n < 3$, entonces no existe C_n. ~~Entonces~~ Y como $n \in \mathbb{N}$ y $n \notin \{3,4\}$, $n \geq 5$.

Supongamos que C_n con $n \geq 5$ fuera un grafo junta. Por lo tanto, existen G_1 y G_2 que contienen solamente algunos nodos y ejes de C_n que forman C_n cuando se los juntan. Sabemos que cada nodo de C_n tiene grado 2. Con esto sabemos que la máxima cantidad de aristas que podemos sumarle a cada nodo de G_1 y G_2 cuando los juntemos será 2. Asimismo, esto nos dice que la máxima cantidad de nodos tanto de G_1 como de G_2 será 2, porque si alguno tuviera más, deberíamos agregar a cada nodo del otro una cantidad mayor a 2 de aristas. Pero $V_{C_n} = V_1 \cup V_2$, y entonces $|V_{C_n}| \leq 4 \Rightarrow n \leq 4$. ABS! que vino de suponer que C_n era un grafo junta.

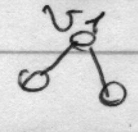
c) P_n es grafo junta $\Leftrightarrow n \in \{2,3\}$

$\boxed{n=2 \Rightarrow P_n \text{: grafo junta}}$ Si tomamos $G_1 = \{v_1\}$ y $G_2 = \{v_2\}$,



vemos que todos los vértices de uno están conectados con los vértices del otro en P_2, y por lo tanto este es su grafo junta.

$\boxed{n=3 \Rightarrow P_3 \text{: grafo junta}}$ Si tomamos $G_1 = \{v_1\}$ y $G_2 = P_3 - v_1$,



vemos que se cumple que su grafo junta es P_3.

LU: 349/16

Nro. de orden: 274	Ejercicio 2 (cont.)	Hoja 3 Eric Brandwein

<u>P_n: grafo junto $\Rightarrow n \in \{2, 3\}$</u>) Si P_n es un grafo junto, ~~...~~ n no podría ser igual a 1, porque entonces no podríamos partir el conjunto de nodos en dos subconjuntos no vacíos.

Como $n \in \mathbb{N}/\{0\}$, $n \geq 2$. Veremos que $n \not> 3$.

Si $n=4$, P_n sería así: $v_1 - v_2 - v_3 - v_4$. Como cada nodo tiene como máximo grado 2, y por lo dicho en el ejercicio b), G_1 y G_2 deben tener como máximo 2 nodos cada uno. Como P_n tiene 4 nodos, G_1 y G_2 deberán tener exactamente 2 cada uno. Pero, a su vez, hay por lo menos 1 nodo de grado 1, ~~y entonces~~ que estará o en G_1 o en G_2, y por lo tanto el subconjunto en el que no esté éste nodo deberá tener como máximo 1 nodo. Esto no es posible, y por lo tanto $n \neq 4$.

Para $n \geq 5$ ocurre una cosa muy parecida a lo ocurrido en el ejercicio b). Como cada nodo de P_n tiene como máximo grado 2, G_1 y G_2 deberán tener como máximo cada uno 2 nodos. Como hay más de 4 nodos, esto no se puede lograr, y entonces P_n no es un grafo junto.

Finalmente, vemos que $n \in \{2, 3\}$, ya que $n \not< 2$ y $n \not> 3$, y $n \in \mathbb{N}/\{0\}$.

d) $K_{p,q}$ es un grafo junto para todo $p, q \in \mathbb{N}/\{0\}$.

Por definición de $K_{p,q}$, ~~existen p nodos que todo~~ existe un conjunto de p nodos, cada uno conectado mediante una arista a cada nodo de otro subconjunto de q nodos.

Si tomamos ~~$G_1 = K_{p,q}$~~ Q como el conjunto de los q nodos y P el de los p nodos, y tomamos $G_1 = K_{p,q} - Q$ y $G_2 = K_{p,q} - P$, veremos que su grafo junto es $K_{p,q}$, ya que al juntarlos todos los nodos de cada uno son conectados entre sí por una arista.

e) Solamente si h = 1 el $\overbrace{\text{árbol binario completo}}^{ABC(h)}$ de altura h será un grafo junto, tomando el árbol trivial como de altura 0.

$\boxed{h=1 \Rightarrow ABC(h)\text{: grafo junto}}$ Como el ABC(1) es isomorfo a P_3, y como vimos que P_3 es un grafo junto, ABC(1) es un grafo junto.

$\boxed{ABC(h)\text{: grafo junto} \Rightarrow h=1}$ Como fue visto en clase, un ABC de altura h tendrá $2^{h+1}-1$ nodos. Además, el máximo grado de cada uno de estos nodos será 3, porque como máximo tendrá dos hijos y un padre.
Si $h \geq 2$, $2^{h+1}-1 \geq 2^{3}-1 = 7$. Para que ABC(h) sea un grafo junto, G_1 y G_2 deberán tener como máximo 3 nodos, porque el máximo grado de un nodo es 3 y por lo visto en el ejercicio b). Pero el ABC(h) tendrá por lo menos 7 nodos, lo que hace imposible encontrar una partición con máximo 3 nodos cada subconjunto, y por lo tanto h no puede ser mayor a 1. Como 1 es el único ~~nodo~~ número natural que nos queda, h=1.

W: 349/16

Nro. de orden 274	Ejercicio 3	Hoja 4 Eric Brandwein

a)

```
      def convertir_a_predecesores(listas_de_sucesores):
O(1)      n = len(listas_de_sucesores)
O(n)      listas_de_predecesores = lista_de_listas(tamaño=n)
          for i ∈ [1...n]:
O(1)          lista_actual = listas_de_sucesores[i]
O(m)          for j ∈ [1...len(lista_actual)]:
 +  O(1)          sucesor = lista_actual[j]
 n  O(1)          listas_de_predecesores[sucesor].agregar(i)
              end for
          end for

O(1)      return listas_de_predecesores
      end def
```

El algoritmo recorre la lista de listas de sucesores, y por cada sucesor agrega a ~~una~~ la lista de predecesores del sucesor el predecesor, que corresponde al índice de la lista actual de sucesores. Como se recorren todas las listas, se agrega un predecesor por cada sucesor, y entonces en la nueva lista se encuentran todos los ejes.

Complejidad:

- Para la longitud de la lista de sucesores podemos utilizar una estructura que identifique la longitud en $O(1)$, como ser una lista basada en un array.
- Para crear una lista de tamaño n podemos asumir que vamos agregando de a un elemento a una estructura con inserción en $O(1)$ amortizado, como ser una lista basada en array. Por lo tanto, la complejidad de ésta línea es $O(n)$.
- Para recorrer la lista de listas de sucesores necesitaremos por lo menos n operaciones, ya que tiene n elementos. A su vez, por cada elemento de cada lista debemos hacer operaciones

de complejidad $O(1)$. Como hay un elemento por eje en el grafo, y m ejes, habrá m elementos totales, y entonces recorrerlos todos tendrá complejidad $O(m)$. La complejidad de todo el bucle interno nos queda entonces en $O(n+m)$.

- Devolver el valor final tendrá complejidad $O(1)$.

Sumando todas las complejidades, nos queda una complejidad total de $O(1+n+n+m+1) = O(n+m)$.

b) ~~Para cada~~ Si invertimos cada eje, su predecesor pasa a ser su sucesor, y viceversa. Por lo tanto, la lista de sucesores de un nodo pasa a ser la de sus predecesores. Si ~~de eso~~ por cada elemento de esa lista agregamos en las listas de sucesores del elemento al nodo v, habremos traducido los ejes de un formato al otro. Si hacemos lo mismo por cada uno de los nodos, habremos convertido las listas de predecesores del grafo invertido en listas de sucesores. Como la lista de predecesores del grafo invertido es igual a la lista de sucesores del grafo original, hace falta solamente llamar a la función del ejercicio a) con la lista de listas de sucesores original. Y, como la función tiene complejidad $O(n+m)$, la complejidad total es la misma.

c) Como ya dijimos en el ejercicio b), la lista de predecesores del grafo invertido es igual a la lista de sucesores del grafo original. Como se nos pide calcular la lista de predecesores del grafo invertido, lo único que tenemos que hacer es devolver la lista de sucesores original, cosa que se puede hacer en $\Theta(1)$, que es estrictamente menor que $\Theta(m+n)$.

LU: 349/16

Nro. de Orden	Ejercicio 4	Hoja 5
274		Eric Brandwein

a) En clase vimos lo que llamamos la propiedad Min-Max, que dice que si T es AGM de un grafo G, entonces para cada par de nodos (u, v) de G, el camino entre u y v en T tiene aristas $\{a_1, a_2, \dots, a_i\}$, tal que el mayor peso de una de esas aristas no supera al peso de la arista de mayor peso de cualquier otro camino entre u y v en G.

Si tenemos a $e = (u, v)$ un eje de T, este eje ya conforma un camino entre u y v, y cualquier otro camino en G de u a v ~~tendrá como peso de la arista de peso máximo a un número mayor o igual al peso de e. Como todas las aristas de G son no negativas, el camino alternativo tendrá como peso mínimo el peso de e, y por lo tanto e conformará un camino mínimo entre u y v~~ y como $e \in T$, este camino es mínimo por la propiedad Min-Max. es la propiedad Min-Max no dice que eso es mínimo.

¿ Esto en el argumento ¿Por qué lo tachaste?

b) La propiedad vista no tiene como condición que los pesos de las aristas sean no-negativos, y por lo tanto el mismo argumento anterior vale para grafos con aristas de peso negativo. ~~[illegible]~~ No, el enunciado es falso si se permiten pesos negativos. La propiedad Min-Max vale, pero eso no implica por sí sola la propiedad del enunciado.

LU: 349/10

Nro. de orden: 274	Ejercicio 5	Hoja 6 Eric Brandwein

$f(2,4) = f(2,3) + d(3,4)$

$f(\cancel{1},\cancel{5}) = \cancel{\ldots}\ \min(\{ f(\overset{4,1}{\cancel{5,4}}) + d(1,5), f(\overset{4,2}{\cancel{5,4}}) + d(2,5), f(\overset{4,3}{\cancel{5,4}}) + d(3,5) \})$

```
def f(i,j){
  si i>j{
    return f(j,i)
  }sino{
    si(j-1)>i{
      return f(i,j-1) + d(j-1, j)
    }sino{
      // j-1=i
      minimo=+∞;{
      for(l ∈ [1..i){
          minimo=min(minimo, f(i,l)+d(l,j))
      }
      return minimo
    }
  }
}
```

El pseudocódigo de la f sugerida en el enunciado podría ser el anterior. Veremos si es cierto.

Si $i>j$, ~~por lo dicho en la~~ se llama a la misma función con los argumentos invertidos. Esto es correcto por ~~toda~~ lo dicho en la sugerencia. Que este en la sugerencia no implica que no haya que justificarlo.

Si $j-1>i$, veremos que $f(i,j) = f(i,j-1) + d(j-1,j)$, siendo $d()$ la función de distancia entre los puntos de correspondiente ~~número~~ índice. ~~Asumimos que f(i,j) calcula la longitud mínima~~

Partamos el camino que calcula $f(i,j)$ en dos: por una parte el camino descendiente de p_i a p_1, y por el otro el ascendiente

de p_1 a p_j. Como el camino de p_i a p_1 no puede pasar por los p_l con $i < l \leq j$, y el camino en conjunto debe pasar por los dos, éstos puntos deben estar en el camino p_1 a p_j. Como un l posible es el $j-1$, p_{j-1} debe estar en ese camino. Y, como tanto p_{j-1} y p_j están en el camino p_1 a p_j, y p_j es el más cercano en x a p_{j-1} con x mayor, la distancia entre los dos debe ser parte del camino de p_1 a p_j. Entonces, el camino que calcula $f(i,j)$ será el mismo que calcula $f(i,j-1)$ si le agregamos la distancia entre $j-1$ y j.

Siguiendo con el pseudocódigo, nos queda el caso en el que $j-1 = i$. En este caso, encontramos el mínimo de los $f(i,k) + d(k,j)$ tal que $k \in [1..i)$. Veremos que esto es correcto.
En $f(i,j)$, el camino de p_1 a p_j debe tener un ~~...~~ punto p_m que sea el anterior a p_j. El camino de $f(i,j)$, entonces, es el mismo que el de $f(i,m) + d(m,j)$, con $m \neq i$. Como $m \neq i$ y $m \neq j$, para encontrar éste m nos debemos fijar en todos los k de 1 a $i-1$, lo que es lo mismo, de 1 a $j-2$. Cuando encontremos el mínimo camino pasando por algún p_k mientras se asciende, le sumaremos la distancia de p_k a p_j, y tendremos el camino mínimo ~~de~~ calculado por $f(i,j)$. ¿Por qué se conviene tomar el mínimo y no otro?

Gracias a ésto, podemos escribir el código del algoritmo que pide el enunciado:

```
def viajante(cant_puntos){  minimo = +∞
    for i ∈ [1..cant_puntos]{
        minimo = min(minimo, f(i, cant_puntos) + d(i, pn))
    }
    return minimo
}
```

¿Complejidad espacial?

Gracias a la memoización, calculamos solamente $O(n)$ por ~~como~~ cada par de índices i, $i+1$, o sea, $O(n^2)$ total, y también calculamos una suma por cada par de índices cualquiera, o sea, $O(n^2)$. ~~...~~ Como al final hacemos comparaciones por mínimo solo n veces, tenemos una complejidad total de $O(n^2)$